WE CLAIM:

1. A method for avoiding the stalling of long immediate data instructions in a pipelined digital processor core having at least fetch, decode, and execution stages, comprising;

identifying, within said pipeline, at least one instruction containing long immediate values;

determining whether said at least one instruction has merged when said at least one instruction is in said decode stage of said pipeline; and

preventing said core from halting before said at least one instruction has merged.

2. The method of Claim 1, wherein said act of determining comprises examining merge logic operatively coupled to said decode stage of said core to determine if a valid merge signal is present.

3. The method of Claim 2, wherein said act of identifying at least one instruction comprises identifying an instruction selected from the group comprising (i) load immediate instructions, and (ii) jump instructions.

4. A digital processor core, comprising:

an instruction pipeline having a plurality of stages;

an instruction set having at least one instruction with multiple word long immediate values associated therewith;

core logic adapted to selectively treat said at least one instruction with said multi-word long immediate values as a single instruction word, said core logic preventing stalling of said core before processing of said at least one instruction has completed.

5. The core of Claim 4, wherein said at least one instruction comprises an opcode and immediate data, said opcode and immediate data having at least one boundary there between, and said core is prevented from stalling on said at least one boundary.

6. The core of Claim 5, wherein said instruction set further comprises a base instruction set and at least one extension instruction, said extension instruction being adapted to perform at least one function not defined within said base instruction set.

7. The core of Claim 6, further comprising extension logic adapted to execute said at least one extension instruction.

8. A method of reducing pipeline delays within a pipelined processor, comprising:

providing a first instruction word;

providing a second instruction word; and

defining a single large instruction word comprising said first and second instruction words;

processing said single large word as a single instruction within said processor, thereby preventing stalling of the pipeline upon execution of said first and second instruction words.

9. The method of Claim 8, wherein the acts of providing said first and second instruction words comprises providing an instruction having at least one long immediate value.

10. The method of Claim 9, wherein the act of providing said instruction having said at least one long immediate value comprises providing an instruction opcode within said first instruction word, and said at least one long immediate value within said second instruction word.

11. The method of Claim 9, wherein the act of processing comprises:

determining whether said first and second instruction words have merged within said pipeline; and

if said first and second words have not merged, preventing said pipeline from stalling on the boundary between said first and said instruction words.

12. A method of processing instruction words within a digital processor having an instruction set comprising a plurality of instruction words, a pipeline with at least first, second, and third pipeline stages, and a program counter adapted to identify at least one address in program memory space, comprising;

providing a program having a plurality of instruction words, including a first instruction word, in said program memory space, said first instruction word resulting in a stall of said pipeline when executed;

inserting said first instruction word into said first pipeline stage; and

delaying decode of said first word until said second stage.

13. The method of Claim 12, wherein the act of inserting comprises inserting said first instruction into said pipeline between other instruction words, and the act of decoding comprises changing the program counter to the memory address value of said first instruction once said first instruction has been decoded in said second pipeline stage.

14. The method of Claim 13, further comprising providing an extension instruction within said instruction set, said extension instruction adapted to perform a predetermined operation when executed on an extension logic unit of said processor.

15. The method of Claim 14, wherein the act of providing an extension instruction comprises providing an instruction adapted for Viterbi decode.

16. The method of Claim 12, wherein said act of inserting comprises disposing said first instruction within a delay slot within said pipeline.

17. A pipelined digital processor, comprising:

a pipeline having instruction fetch, decode, execute, and writeback stages;

a program memory adapted to store a plurality of instructions at addresses therein;

a program counter adapted to provide at least one value corresponding to a at least one of said addresses in said memory;

decode logic associated with said decode stage of said pipeline;

an instruction set comprising a plurality of instructions, said plurality further comprising at least one breakpoint instruction; and

a program comprising a predetermined sequence of at least a portion of said plurality of instructions, and including said at least one breakpoint instruction, said program being stored at least in part in said program memory;

wherein the decode of said at least one breakpoint instruction during execution of said program occurs after said instruction fetch stage using said decode logic, and said wherein said program counter is reset back to the memory address value associated with said breakpoint instruction after said breakpoint instruction is decoded.

18. The processor of Claim 17, further comprising an extension logic unit adapted to execute one or more extension instructions.

19. The processor of Claim 18, wherein said instruction set further comprises at least one extension instruction, said at least one extension instruction adapted to perform a predetermined function upon execution within said extension logic unit.

20. A method of debugging a digital processor having a multi-stage pipeline with fetch, decode, execute, and writeback stages, a program memory, a program counter adapted to provide at least one address within said memory, and an instruction set stored at least in part within said program memory, said instruction set including at least one breakpoint instruction, comprising;

providing a program comprising at least a portion of said instruction set and at least one breakpoint instruction;

running said program on said processor;

decoding said at least one breakpoint instruction during program execution at said decode stage of the pipeline;

executing the breakpoint instruction in order to halt operation of said processor;

resetting said program counter to the memory address value associated with said breakpoint instruction; and

debugging said processor at least in part while said processor is halted.

21. The method of Claim 20, whereinsaid instruction set includes at least one extension instruction, said at least one extension instruction adapted to perform a predetermined function upon execution within said processor, said act of providing a program further comprises providing said at least one extension instruction therein, said method further comprising executing said at least one extension instruction during said dedubbing.

22. A method of enhancing the performance of a digital processor design, said processor design having a multi-stage instruction pipeline including at least instruction fetch, decode, and execution stages, an instruction set having at least one breakpoint

instruction associated therewith, a program memory, and a program counter controlled at least in part by pipeline control logic, the method comprising:

providing a program comprising at least a portion of said instruction set, said at least portion including said breakpoint instruction;

simulating the operation of said processor using said program;

identifying a first critical path within the processing of said program based at least in part on said act of simulating, said critical path including the processing of said breakpoint instruction within said program; and

modifying said design to decode said breakpoint instruction within said decode stage of said pipeline so as to reduce processing delays associated with said first critical path.

23. The method of Claim 22, wherein the act of modifying further comprises adapting said pipeline control logic so that said program counter resets to the memory address value associated with said breakpoint instruction after said breakpoint instruction is decoded within said decode stage.

24. A method of reducing pipeline delays within the pipeline of a digital processor, comprising:

providing a first register having a plurality of operating modes;

defining a bypass mode for said first register, wherein during operation in said bypass mode, said register maintains the result of a first multi-cycle operation therein;

performing a first multi-cycle operation to produce a first result;

storing said first result of said first operation in said first register using said bypass mode;

obtaining said first result of said first operation directly from said register; and

performing a second multi-cycle operation using at least said first result of said first operation, said second operation producing a second result .

25. The method of Claim 24, wherein said multi-cycle operation comprises an iterative scalar calculation, said method further comprising performing the acts of storing, obtaining, and performing for said second result of said second operation, and a

plurality of subsequent results from respective subsequent operations, wherein the result of a given operation is stored in said first register using said bypass mode, and subsequently obtained from said register for use in the next subsequent iteration of said calculation.

26. A processor core, comprising:

a multi-stage instruction pipeline having at least fetch, decode, and execute stages;

an instruction set having at least one multi-cycle instruction and at least one other instruction subsequent thereto; and

a first register disposed within the execute stage of said pipeline, said first register having a bypass mode associated therewith, said bypass mode adapted to:

(i) retain at least a portion of the result of the execution of said at least one multi-cycle instruction within said execute stage; and

(ii) present said result to said at least one other instruction for use thereby.

27. The processor core of Claim 26, wherein said first register is further adapted to latch source operands to permit fully static operation.

28. The processor core of Claim 26, wherein said at least one multi-cycle instruction comprises two sequential data words, the first of said data words comprising at least opcode, and the second of said data words comprising at least one operand.

29. The processor core of Claim 28, further comprising core logic adapted to selectively treat said at least one multi-cycle instruction with said data words as a single instruction word, said core logic preventing stalling of said core before processing of said at least one instruction has completed.

30. The processor core of Claim 28, wherein said instruction set further comprises at least one extension instruction, said at least one extension instruction being adapted to perform a predetermined function upon execution thereof by said core.

31. The processor core of Claim 30, further comprising an extension logic unit adapted to execute said at least one extension instruction.

32. A method of operating a data cache within a pipelined processor, said pipeline comprising a plurality of stages including at least decode and execute stages, at

least one execution unit within said execute stage, and pipeline control logic, said method comprising:

providing a plurality of instruction words;

introducing said plurality of instruction words within said stages of said pipeline successively;

allowing said instruction words to advance one stage ahead of the data word within said data cache;

examining the status of said data cache; and

stalling said pipeline using said control logic only when a data word required by said at least one execution unit is not present within said data cache.

33. The method of Claim 32, further comprising:

making said data word available to said execution unit; and

updating the operand for the instruction in the stage prior to said execute stage.

34. The method of Claim 33, wherein the act of updating comprises updating the operand in the decode stage of said pipeline.

35. A pipelined digital processor, comprising:

a pipeline having instruction fetch, decode, execute, and writeback stages;

storage means adapted for storing a plurality of instructions at addresses therein;

address generation means for providing at least one value corresponding to a at least one of said addresses in said storage;

means for decoding an instruction word, said means for decoding associated with said decode stage of said pipeline;

an instruction set comprising a plurality of instructions, said plurality further comprising at least one instruction means for halting operation of said processor pipeline; and

a program comprising a predetermined sequence of at least a portion of said plurality of instructions, and including said at least one instruction means for halting, said program being stored at least in part in said storage means;

wherein the decode of said at least one instruction means during execution of said program occurs after said instruction fetch stage using said means for

decoding, and said wherein said address generation means is reset back to the address value associated with said instruction means after said instruction means is decoded.

36. A processor core, comprising:

a multi-stage instruction pipeline having at least fetch, decode, and execute stages;

an instruction set having at least one multi-cycle instruction means and at least one other instruction subsequent thereto; and

means or storing disposed within the execute stage of said pipeline, said means for storing having a bypass means associated therewith, said bypass means adapted to perform the steps comprising:

(i) retaining at least a portion of the result of the execution of said at least one multi-cycle instruction means within said execute stage; and

(ii) presenting said result to said at least one other instruction for use thereby.

37. The procesor core of Claim 36, wherein steps (i) and (ii) are performed repetitively by said means for storing and said bypass means.

38. A method of synthesizing the design of an integrated circuit, said design including a pipelined processor having optimized pipeline performance:

providing input regarding the configuration of said design, said configuration including at least one optimized pipeline architectural function;

providing at least one library of functions, said at least one library comprising descriptions of functions including that of said at least one pipeline architectural function;

creating a functional description of said design based on said input and said at least one library of functions;

determining a design hierarchy based on said input and at least one library;

generating structural HDL and a script associated therewith;

running said script to create a synthesis script; and

synthesizing said design using synthesis script.

39. The method of Claim 38, wherein the act of providing input regarding the said at least one optimized pipeline architectural function comprises:

describing at least one multi-word instruction comprising a first opcode word and a second data word; and

specifying that said instruction is non-stallable on the boundary between said first and second words during execution thereof.

40. The method of Claim 38, wherein the act of providing input regarding the said at least one optimized pipeline architectural function comprises:

describing a multi-function register disposed within said pipeline, said register adapted to store the results of the execution of a multi-cycle instruction word within the execute stage of said pipeline; and

specifying that sid result be provided to at least one instruction subsequent to said multi-cycle instruction within said pipleine during operation.

41. The method of Claim 38, wherein the act of providing input regarding the said at least one optimized pipeline architectural function comprises:

describing pipeline control logic adapted to control the operation of said pipeline;

describing at least one execution unit within the execution stage of said pipeline;

describing at least one data cache structure within said design;

specifying that said pipeline control logic be at least partly decoupled from said data cache, thereby allowing the processing of a given instruction within said pipeline to proceed ahead of said data cache; and

further specifying that said pipeline control logic halt said pipeline if a data word required by said at least one execution unit is not present within said data cache.